

# NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



## THESIS

**COPS FOR WINDOWS:  
A SMALL-SCALE, NETWORKED,  
INFORMATION SYSTEM**

by

Lawrence Albert Nathan

September, 1995

Thesis Advisor:  
Co-Advisor:

Timothy Shimeall  
John Falby

Approved for public release; distribution is unlimited.

19960208 116

FINAL QUALITY INSPECTED 1

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September, 1995.	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE COPS FOR WINDOWS: A SMALL-SCALE, NETWORKED, INFORMATION SYSTEM.		5. FUNDING NUMBERS		
6. AUTHOR(S) Lawrence Albert Nathan				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) <p>The existing Computerized On-Line Police System in use by the Naval Postgraduate School Police Department lacks the information storage, retrieval, query, and data-sharing functionality necessary to maintain personnel and related vehicle and citation records of approximately 7,000 people. The flat file system in use by two different users creates data inconsistencies and wastes time and storage resources by trying to duplicate data on two different machines.</p> <p>The approach taken to solve this problem was the application of a modified Waterfall development cycle to create a new database application that would replace the flat-file system. This thesis describes the development effort from analysis through implementation.</p> <p>The resulting application is called COPS for Windows. COPS for Windows allows more than seventy-five people simultaneous access to the same data files while providing different views of the information and maintains data integrity and consistency. The program generates reports such as the weekly docket, suspension letters, and probation lists. Data entry screens and reports are modeled on existing paper documents. COPS for Windows is a network-ready, multi-user, information system with extensive querying, reporting, and storage capabilities to integrate support cost effectively.</p>				
14. SUBJECT TERMS Small-Scale Networked Database.			15. NUMBER OF PAGES 92	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	



Approved for public release; distribution is unlimited.

**COPS FOR WINDOWS:  
A SMALL-SCALE, NETWORKED,  
INFORMATION SYSTEM**

Lawrence A. Nathan  
Captain, United States Marine Corps  
B.S., Virginia Military Institute, 1989

Submitted in partial fulfillment  
of the requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

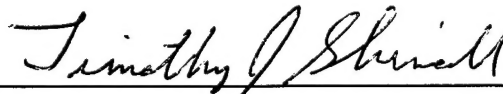
**NAVAL POSTGRADUATE SCHOOL  
September 1995**

Author:



Lawrence A. Nathan

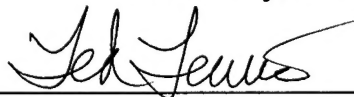
Approved by:



Timothy Shimeall, Thesis Advisor



John Falby, Co-Advisor



Ted Lewis, Chairman

Department of Computer Science



## **ABSTRACT**

The existing Computerized On-Line Police System in use by the Naval Postgraduate School Police Department lacks the information storage, retrieval, query, and data-sharing functionality necessary to maintain personnel and related vehicle and citation records of approximately 7,000 people. The flat file system in use by two different users creates data inconsistencies and wastes time and storage resources by trying to duplicate data on two different machines.

The approach taken to solve this problem was the application of a modified Waterfall development cycle to create a new database application that would replace the flat-file system. This thesis describes the development effort from analysis through implementation.

The resulting application is called COPS for Windows. COPS for Windows allows more than seventy-five people simultaneous access to the same data files while providing different views of the information and maintains data integrity and consistency. The program generates reports such as the weekly docket, suspension letters, and probation lists. Data entry screens and reports are modeled on existing paper documents. COPS for Windows is a network-ready, multi-user, information system with extensive querying, reporting, and storage capabilities to integrate support cost effectively.



## TABLE OF CONTENTS

I. INTRODUCTION .....	1
A. SOFTWARE DEVELOPMENT .....	1
1. Analysis .....	1
2. Design a Solution .....	2
3. Code .....	2
4. Verification and Validation .....	2
5. Operation and Maintenance .....	2
B. DESIGN PARADIGMS .....	2
C. DATABASE DEVELOPMENT .....	3
D. DATABASE APPLICATION DEVELOPMENT .....	4
1. Analysis .....	4
2. Design a Solution .....	4
3. Verification, Validation, and Maintenance .....	4
E. WATERFALL PARADIGM .....	4
F. CONCLUSION .....	5
G. CHAPTER SUMMARY .....	6
II. BACKGROUND .....	7
A. COPS VERSION ONE .....	7
B. THE NEW SYSTEM: COPS for WINDOWS .....	8
C. DESIGN GOALS .....	9
1. Security .....	9
2. Reports .....	9
3. Forms .....	10
4. Queries .....	10
5. Processing .....	10



6. Interface .....	10
D. CONCLUSION .....	11
III. SYSTEM DESIGN .....	13
A. INTRODUCTION .....	13
B. E-R DIAGRAM .....	13
C. E-R $\Rightarrow$ SCHEMA MAPPING .....	14
D. TABLE DESIGN .....	16
E. TABLE CONSTRAINTS .....	19
1. Referential Integrity .....	19
2. Secondary Indexes .....	20
F. IMPLEMENTATION .....	20
G. DATA CONVERSION AND VALIDATION .....	20
1. Ticket Table Conversion .....	21
2. Vehicle Table Conversion .....	21
3. Person Table Conversion .....	21
4. Remove Redundant Data .....	22
5. Remove Orphan Records .....	23
H. INSTALLATION .....	23
IV. SYSTEM VALIDATION .....	25
A. INTRODUCTION .....	25
B. REQUIREMENTS CLASSIFICATION .....	25
C. AGGREGATION .....	26
D. TEST CASE OVERVIEW .....	26
E. SUMMARY OF RESULTS .....	26
V. CONCLUSIONS .....	29

A.	RESULTS .....	29
B.	RECOMMENDATIONS .....	29
C.	FUTURE WORK .....	29
APPENDIX A. USERS MANUAL .....		31
A.	INTRODUCTION .....	31
B.	INSTALLATION .....	31
C.	LOGGING IN .....	32
D.	MAIN MENU .....	32
E.	DATA ENTRY .....	33
F.	TICKET .....	34
	1. New .....	34
	2. Modify .....	34
	3. Suspension .....	34
	4. Probation .....	35
	5. DUI .....	35
G.	REPORTS .....	35
	1. Docket .....	35
	2. Statistics .....	35
	3. Suspension .....	36
	4. Probation .....	36
H.	BROWSE .....	36
I.	DESK JOURNAL .....	37
	1. Open .....	37
	2. Close .....	37
	3. View .....	37
	4. New .....	37
J.	LOGGING OUT .....	38

APPENDIX B. REQUIREMENTS DOCUMENT .....	39
A.    INTRODUCTION .....	39
B.    SYSTEM MODEL .....	39
C.    SYSTEM EVOLUTION .....	41
D.    FUNCTIONAL REQUIREMENTS SPECIFICATION .....	42
1. Reports .....	42
2. Forms .....	42
3. Interface .....	43
4. Security .....	45
5. Scantron .....	45
6. Queries .....	45
7. Validation Tables .....	46
8. Goals .....	46
9. Constraints .....	47
10. Assumptions .....	47
E.    NON-FUNCTIONAL REQUIREMENTS SPECIFICATION .....	47
1. Hardware .....	47
2. Software .....	47
3. Documentation .....	47
4. Training .....	48
5. Installation .....	48
6. Maintenance .....	48
F.    DATABASE REQUIREMENTS .....	48
APPENDIX C. TEST INFORMATION .....	67
A.    TEST REQUIREMENTS .....	67
B.    TEST CASES .....	69

1. Aggregation One: Inspection .....	69
2. Aggregation Two: Execution .....	70
LIST OF REFERENCES .....	75
INITIAL DISTRIBUTION LIST .....	77



## ACKNOWLEDGMENTS

I would like to acknowledge the NPS police department for their support, encouragement, and enthusiasm with this project. In particular I would like to thank the following people for their patience and willingness to try something new: Alex Wills, Gregg Caughran, and Marilyn Owens.

I would also like to thank my thesis advisors: Prof. John Falby for continuing to push (but never too hard), and Prof. Tim Shimeall for lending me his expertise (and his books).

Foremost, I would like to acknowledge the support of my wife whose understanding and patience when I told her I would have to work late or on weekends made those long hours bearable.

## **I. INTRODUCTION**

As the use of Personal Computers (PC's) grows, many small-scale applications become economically justifiable. A small application, one that will be used by twenty-five or fewer people, can be developed timely, efficiently, and inexpensively by employing good software design methods. This small-scale design, or Programming-in-the-Small, is rarely discussed in schools or in books since it is assumed to be trivial. For the large software company with hundreds of programmers and developers this may be true, however for the one-to-four person development team that might be hired as a small business consultant this type of programming should not be taken lightly. As more small businesses step into the computer age this type of programming will become much more of a demand that will need to be filled. This thesis demonstrates how such an application may be built with a case study that steps through the entire development process. The final product is an application for the Naval Postgraduate School (NPS) Police department. It is designed to allow the sharing of the data files on a network server (LAN based), and automates some of their record-keeping and reporting procedures.

### **A. SOFTWARE DEVELOPMENT**

The science of software development has been evolving for many years. It has reached such a formal level that developers now call themselves "Software Engineers." The design process that these engineers step through is similar to other engineering design processes:

#### **1. Analysis**

During the analysis phase the developers and customers work closely together to determine the exact requirements and constraints of the system. These requirements may include nonfunctional requirements such as a specific programming language, and functional requirements such as a menu or help system. Constraints may include things such as time, when it should be completed, and money, how much it should cost. All these constraints and requirements are specified in a language such that both the customers and the developers can clearly understand them and put them into documents such as feasibility study, requirements'

specification, or an acceptance test plan. These documents can then form the basis of a contract between the developers and the customer.

## **2. Design a Solution**

The design phase creates a model of the system on paper. The database requirements and data structures are determined and the system is broken into several modules. Each module can then be either subdivided again or transformed into executable code that can be run and tested. The design phase produces the design specification that would then go to the programmers for coding.

## **3. Code**

This phase turns all of the designed modules into executable code. Unit level testing is performed on each module. When the modules are individually debugged, they can be integrated and compiled into a fully executable program.

## **4. Verification and Validation**

Verification is a thorough testing of the executable code based on the test specification developed in the analysis phase. Validation is a testing of the program based on the design and requirement's specification. In short, verification tests "Are we building the product right?" and validation tests "Are we building the right product?"[BOEH81].

## **5. Operation and Maintenance**

The software is then installed according to the contract. This might include full installation on all hardware and training, or it might just be the delivery of the software and a user's manual, or somewhere in between. The delivery includes product acceptance (or rejection) by the customer and debugging any problems that might not have been discovered earlier during the test phases. Upgrades or improvements to the application might also be included as part of the maintenance cycle or possibly form the basis of the next contract.

## **B. DESIGN PARADIGMS**

These five steps can be organized and arranged in a variety of ways. Some more popular paradigms include: the Waterfall, which goes through each step only once and in order; Exploratory Programming, where a working application is developed quickly and then



is modified to perfection; and Prototyping, which develops an application quickly so the end users can put their hands on it and help develop requirements for the final product.

Although these paradigms can give the implementors a method for solving the problem, how they get there can still vary greatly from project to project. These variances can occur for many different reasons such as what type of project they are working on, how many people are working on that project, what contract requirements there are (reports or presentations), and the actual design method being used (Object Oriented Design or Function Oriented Design). Everyone uses a paradigm in solving his or her problem. Whether it is a single hacker programming at home that does most of this in his head or a large team from a major corporation working on a new operating system, the five previously mentioned steps will be organized so that it will make sense to the person in charge of that project.

### **C. DATABASE DEVELOPMENT**

A database is a collection of related facts that have meaning. It should be logically coherent, represent some aspect of the real world (a person might be represented by a name and a social security number), and should be designed, built, and maintained for some specific purpose. [ELMA94] This case study develops an application whose backbone is a database that tracks people vehicles, tickets, and incidents.

Advances in PC software have allowed the development of full scale Database Management Systems (DBMS) that were previously only available on large mainframe computers. These DBMS's allow users to create their own database applications that can be run as a script on top of the DBMS, or in some cases compiled into an executable file and distributed. The design of this type of application is very similar to the design process for other software development. The analysis phase concentrates on what data needs to be maintained, the characteristics to be used to model this data, and the interface between the user and the data. This phase is a machine independent level and concentrates on a complete understanding of the problem to be solved (conceptual design). The design phase is concerned with reports, forms, interfaces, queries, and specifically the tables that hold the information. This phase includes a mapping from the conceptual design to a schema that is

dependent upon the type of database being used (relational, hierarchical, network . . . ). This mapping is called the logical design. The coding phase consists of the implementation of the tables, forms, reports, queries, and macros determined in the previous step. All of this is done in a language(s) specified by the DBMS.

#### **D. DATABASE APPLICATION DEVELOPMENT**

The database design process and the software development process can be effectively combined and utilized to fill the ever increasing needs of small-scale information systems (SSIS). A SSIS is an application that will be used by one to twenty people and consists primarily with the storage and formatted retrieval of information.

##### **1. Analysis**

This phase contains all of the analysis items mentioned previously from both the software design method and the database design method. Developers work closely with the client, specifically the actual users, to determine the system needs. The output documents from this analysis reflect a system that takes the users skills, needs, and functions into consideration.

##### **2. Design a Solution**

The design phase still creates a model of the system on paper. It includes the logical design of the database and clearly demonstrates the interaction between the user, the interface, and the application. The design phase produces the design specification that then goes to the programmers for coding.

##### **3. Verification, Validation, and Maintenance**

The remaining two phases remain the same as in the application development process. The whole system is tested and delivered. Any problems noted during actual operation, they are fixed or documented.

#### **E. WATERFALL PARADIGM**

The methodology outlined above is commonly known as the Waterfall design method. It is a document-based method that flows from step to step much like a waterfall does. At each phase one or more documents are produced that form the foundation for the next step

in the development cycle. The document(s) is reviewed thoroughly; approval to move on to the next step is given only after all discrepancies in the current phase have been fixed.

This method can have some serious drawbacks. The creation and approval of these documents are very time consuming and can slow the development process. The very nature of the Waterfall process freezes the development at a certain stage, gets validated and verified, then moves on never to return. Development should not work this way. The development process should look both forward and back planning for the next stage and finding problems with the current design. These newfound problems should be taken care of by going back to the stage of the design where it was neglected. Most often however these problems tend to be ignored or hacked together [SOMM92].

For this project, a modified Waterfall method is used. The project's small scale means the design documentation will not be so long or cumbersome as to slow the development process during reviews. By requiring the documentation reviews, the project is ensured a timely completion. The modification to the Waterfall method is simply the allowance of modification of previous stages of the design that will ensure the best design and ultimately the best product. This flow is depicted in Figure 1.

## **F. CONCLUSION**

This thesis does not focus on any particular research question. It is an application of solid software engineering principles and database development to solve a real world problem. The application forms a case study of programming in the small of a data management system. With this combined process as a guideline and a good DBMS that can be used for implementation, programming in the small, although not trivial, is much easier.

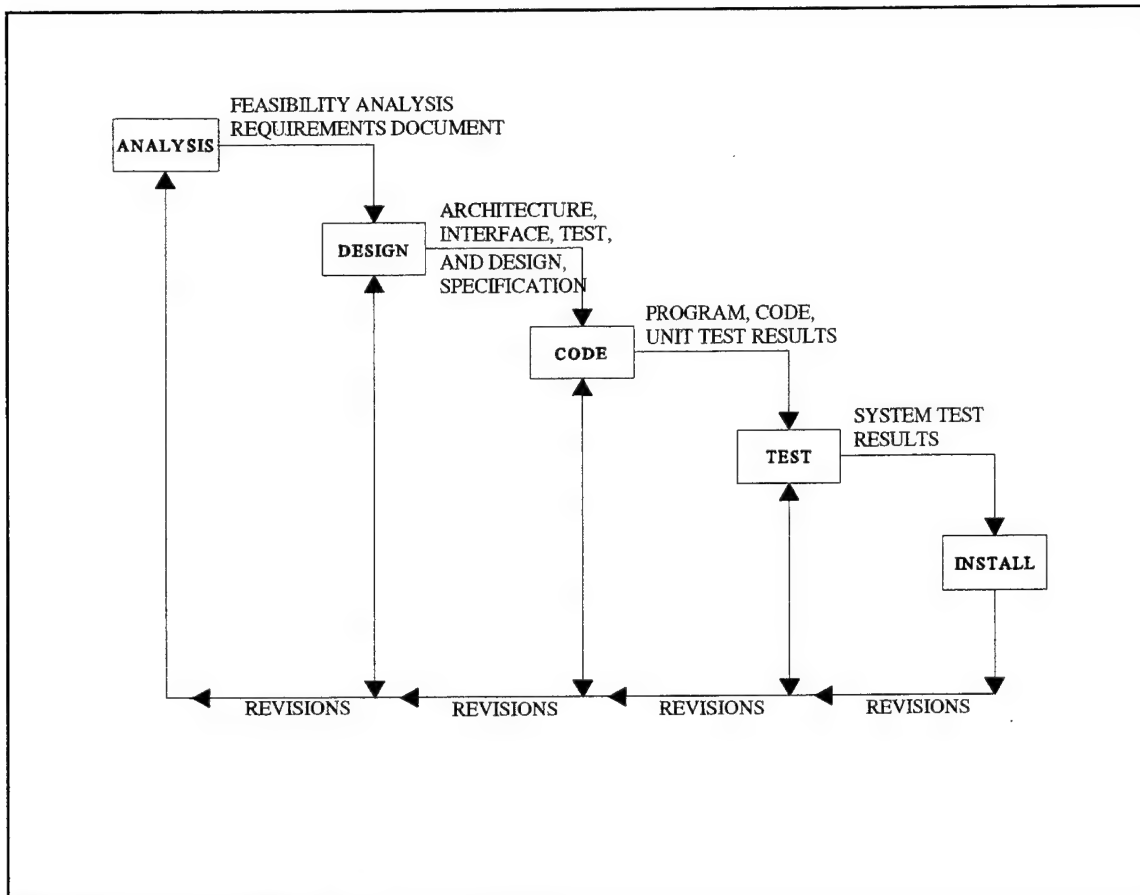


Figure 1. Modified Waterfall Method

## G. CHAPTER SUMMARY

Chapter II presents a background of the current Computerized on-line Police System (COPS) for vehicle registration and presents some high level goals the new application achieves. Chapter III presents some database design documents such as Entity-Relationship (E-R) Diagrams, and the Schema mapping. Chapter IV discusses system validation and verification, the methods used and the test results. Chapter V is a summary and conclusion that discuss results and possible future work. Appendix A is the User's Manual for the developed application. Appendix B contains the remainder of the design documentation, in particular the Requirements Document. Appendix C contains testing documents.

## **II. BACKGROUND**

### **A. COPS VERSION ONE**

The Computerized On-Line Police System, COPS, project started in May 1986. The original version was developed by the Naval Computer and Telecommunications Station in San Diego, California. This version of COPS was developed using dBaseII© ver. 2.41 by Ashton Tate. By September 1991 this version was converted to dBase III plus© also by Ashton Tate, then compiled using Clipper© of Nantucket Corp. [NAVA91]

COPS consists of six database files. Two files are used for security to ensure only authorized users can view or modify data. A third file is used for validation to ensure the consistency of certain field entries such as state/country codes. The three remaining files are the heart of the application and contain the data that represents people, vehicles, tickets, and decals.

The system requirements for COPS are minimal when compared to modern computing standards. The minimum hardware requirements are a Zenith Z248 microcomputer (Intel 286 compatible), a 5 1/4 inch floppy drive and a 20 Mb hard drive. If printed reports are needed, then a printer with a 132 character-width is required. The software requirements are MS DOS ver. 3.2 or greater. [NAVA91]

This version of COPS is currently in use by the NPS Base Police. It was originally used on two separate PC's at different locations. The Vehicle Registrar worked with his own data, entering people and vehicles, while the Traffic Administrator worked with separate data and input ticketing information. This is known as the File Processing method. This method allows for different file sets for each user at different locations. This also has the drawbacks of inconsistencies from data input errors, and wasted storage space and wasted time due to maintenance of redundant data. The File Processing approach tends to be very inefficient and inaccurate for multi-user needs.

This version of the COPS application is insufficient in several areas. The canned transactions are not powerful enough to satisfy most of their query needs. The queries are

based almost exclusively on a person's social security number. This is ineffective if the only information that the clerks have is the vehicle's decal number (a very commonplace occurrence for the ticket administrator). It does not have the reporting facilities to generate the required reports or the statistical ability to analyze the data collected. Monthly and yearly reports are required for analysis of the type and location of tickets issued. Driver suspension reports are required monthly so they may be given to the police officers. Currently this is all done manually and takes several hours weekly. Several other manual operations that could have been incorporated were not, such as pet and weapon registration or blotter/incident reports. The greatest perceived problem is its lack of flexibility. There is no way to improve on this version. The clients must have another version created specifically for them.

## **B. THE NEW SYSTEM: COPS for WINDOWS**

The new version is designed specifically to answer these shortcomings. It reduces the inefficiencies created with multiple data sets; decreases time spent on manual processing of transactions such as entering new vehicle-registration or ticket information; generates the necessary reports that are currently done manually; incorporates the operations done manually so they may be automated; and is designed for the future so that possible growths in the Police Departments will not outgrow the application.

To achieve this, the new system is created with a network and multiple users in mind. It was developed to be a local program (an application that runs locally on the user's PC) that accesses shared data through a network file server. It was designed in a window's environment to provide an easy-to-use graphical user interface (GUI). Data consistency is maintained by referential integrity. Concurrency control is maintained with file/record locking. By working closely with the actual users, efficient reports and transactions have been designed resulting in a dramatic increase in productivity gains.

COPS for Windows was designed specifically for the NPS Base Police. In particular, it was designed for the vehicle registration clerk and the traffic administrator. It was designed to be used with minimal training on the application itself (it is expected that the individual knows his or her own job). For security purposes, a system manager is necessary to set up

user accounts. These accounts are necessary to ensure casual users do not gain access to the files through the application and perform unwanted data manipulation. It is designed to be run locally on a PC through a Windows environment with shared data files accessed through a network file server. A printer supported by windows is necessary for the reports, and a Scantron Model 8200 optical mark reader is an optional piece of equipment that can provide an alternative to entering vehicle registration data by keyboard.

### **C. DESIGN GOALS**

These requirements and design goals were determined through interviews with the clients during the analysis phase. The goals are presented here in a very broad sense. They are described in much greater detail in the actual design documentation found in Appendix B.

#### **1. Security**

The security requirements are twofold. First is a requirement for security at the user level to help ensure data integrity. A security level is assigned to each user. Only users with the appropriate access level are allowed to modify data, all other users can only read. Second, the data files themselves must be protected. Since the data will be on a shared file service, unauthorized users may gain access to the data files. Although they probably could not access the information through the application, they could use another database application to perform some damage to the data. This requirement only applies to the table containing the user passwords. The clients decided to rely on the network security in place rather than have all of the tables encrypted.

#### **2. Reports**

A database application is of little to no use if the users cannot retrieve the information stored within. Most of the reports are generated by the Traffic Administrator. These reports include ticket analysis reports (what type of ticket was issued and where), suspension reports (who is not allowed to drive on base), and letters of suspension (when a person receives two or more traffic violations).

### **3. Forms**

Any application should be easy for the clients to use. Not only should it be easy to extract information in useful reports but entering information into the database should also be easy. Forms are used to enter information. To make the application easy and useful, these forms are designed to look just like the paper forms the clerks currently use. For the Traffic Administer, the forms look like tickets. For the Vehicle Registrar, the forms look like the vehicle registration form.

### **4. Queries**

Processing of the maintained information is also important. Queries should be based on the information that the user has in hand. The Traffic Administrator's most common task is entering parking tickets. A parking ticket is usually written when there is no driver of the vehicle present. Because of this, the Ticket Administrator must search through the database using the Department of Defense (DOD) decal as the index to retrieve the remaining information necessary to fill the ticket. It would be absolutely no good to have the query based on the recipient's Social Security Number (SSN), since that information is not available.

### **5. Processing**

The processing of information is necessary not only for the reports but also for maintaining the usability of the database. It does not do any good to maintain information on a person that has moved away. The application can do a certain degree of information processing including the removal of personnel files and all other dependent child records when their rotation date passes.

### **6. Interface**

Above all these other categories of goals is the user interface. There must always be some tradeoffs for different requirements. For example, security, speed, and ease of use are commonly traded. The more secure a system the slower it gets and usually more cumbersome to use. The interface must strike a balance among all the different requirements and goals. It must walk several fine lines without crossing any or the whole application can be declared a failure. All of the capabilities of the application must be easy to access or they will not be



used. If certain features are not in use then they are being done in another fashion usually manually and thus wasting valuable time and money.

#### **D. CONCLUSION**

Great care must go into the analysis of the requirements and the overall design of the application. It should be easy to use. The inputs, outputs, documentation, and controls should be convenient, straightforward, and natural. It should satisfy human needs and fulfill human potential. It should help do the job, not overdo the job. It should take the user's skills, needs, and functions into consideration and be reflected in the reports, forms, controls, and documentation of the application. [BOEH81] Only then can the application be called a success!



### **III. SYSTEM DESIGN**

#### **A. INTRODUCTION**

The first step in the design of the database is the creation of the Entity-Relationship (E-R) diagram. This diagram is then mapped to a schema that is used for the creation of the tables used in the application. As the tables are being built, validity checks such as default values, integrity constraints such as referential integrity, and secondary indexes are added as well. Once the tables are built, the application interface is created, and then the reports. This chapter steps through the design of the tables.

#### **B. E-R DIAGRAM**

The E-R diagram is an implementation-independent representation of the data to be modeled. It describes all the information that is to be stored within the database and how that information is interrelated. It is nontechnical, therefore the diagram may be used to communicate with the end-users to ensure that all database requirements are captured in the design without any conflicts [ELMA94].

Each independent data item such as a person or vehicle is called an entity. An entity is represented by a unique name and a list of attributes. The attribute values completely describe a specific instance of an entity. For example, PERSON is an entity. Its attributes might include NAME, ADDRESS, and SSN. An instance of the PERSON entity might be NAME: JOHN DOE, ADDRESS: 321 BAKER ST., SSN: 999-99-9999. The minimum number of attributes required to identify uniquely an entity instance is called a key. For the PERSON example, SSN would be a key attribute because every person has a unique SSN. Several attributes may be combined to form a composite key such as NAME and BIRTHDATE (this assumes that no two people born on the same day have the same name). If an entity has more than one key, they are called candidate keys. If an entity does not have any key attributes, it is called a weak entity type. Weak entity types have partial keys that when taken in conjunction with the owning entity's key will uniquely identify it. [ELMA94]

Entities are represented by a rectangle in the E-R diagram. Relationships among entities are depicted with diamonds. Characters are used to describe the participation constraints of the relationships. Attributes are represented as circles within the E-R diagram. Because of its complexity, the attributes are sometimes left off the E-R Diagram and then listed separately. The COPS for Windows E-R diagram is listed in Figure 2 and the attributes are listed separately in Appendix B Section F.

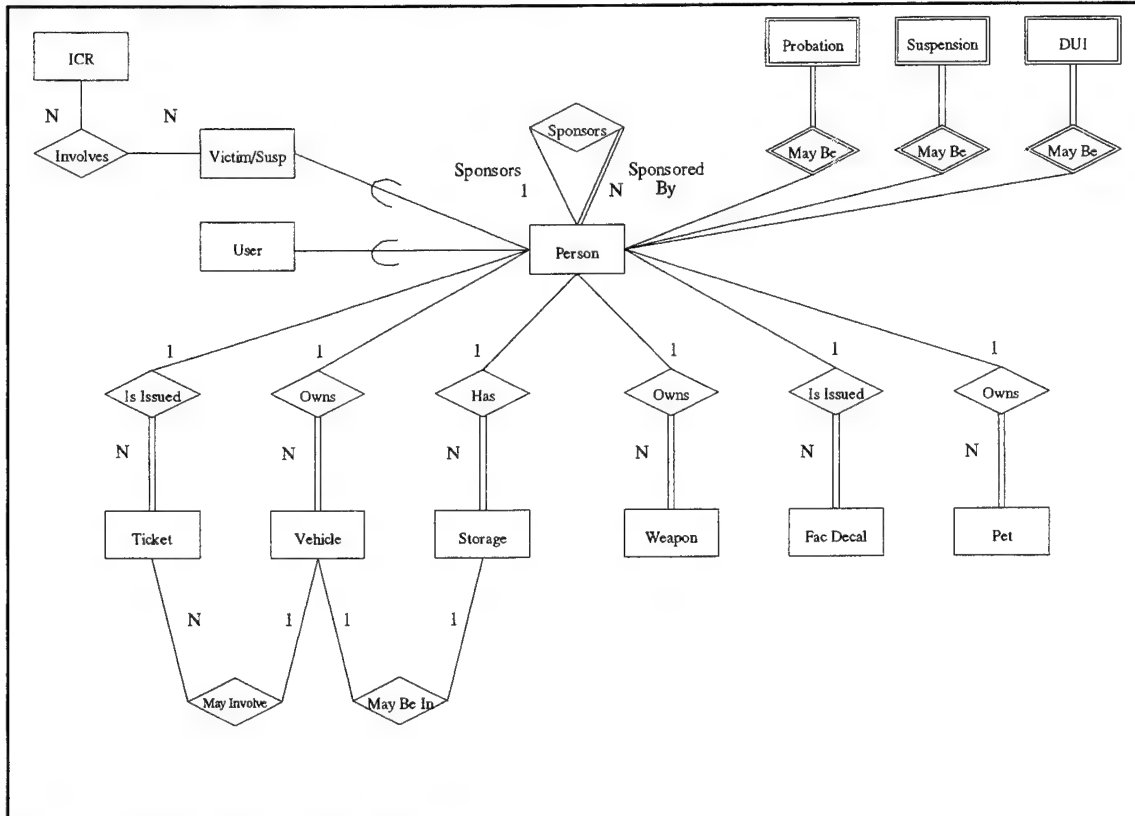


Figure 2. Entity Relationship Diagram

### C. E-R $\Rightarrow$ SCHEMA MAPPING

Once the E-R diagram has been refined and approved, it is then converted into a Schema. The Schema is a closer representation of the actual DBMS tables to be built. Primary keys are chosen for each entity. Relations are removed and replaced with foreign keys. Complex attributes and relations (multi-valued or composite attributes and weak entity

types) are mapped into simple atomic attributes that can be built into a table. A very simple example of an E-R diagram and its schema mapping is shown in Figure 3.

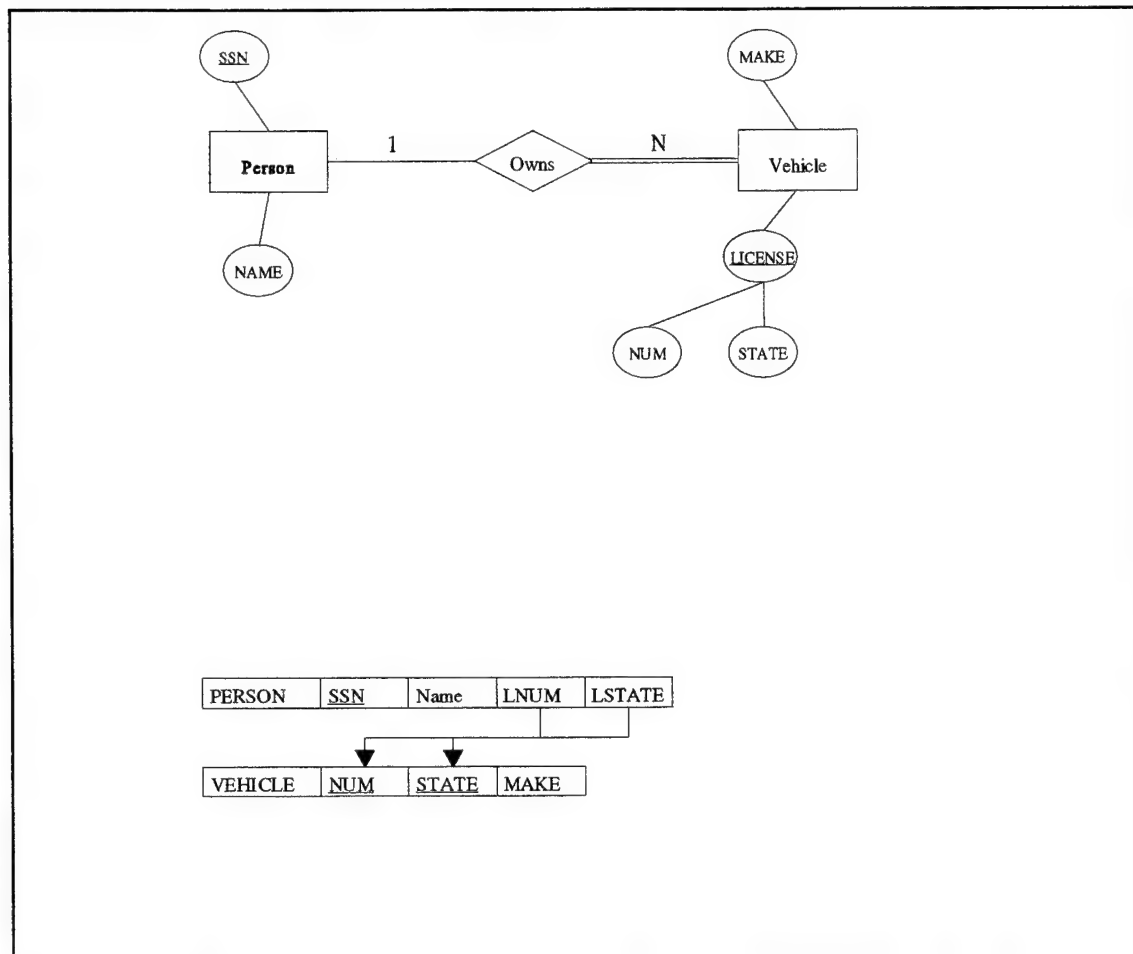


Figure 3. Sample E-R Diagram and Schema

The schema is then normalized (redundant data and relationships are removed) and used to build the tables that will be storing the data. A schema for the person, vehicle, and ticket entities is shown in Figure 4.

PERSON

SSN	FNAME	LNAME	MI	STREET	CITY	STATE	ZIP
GRADE	RANK	SMC#	DUTY STATION			LIC_NUM	
LIC_STATE		BOS	FAC/STAFF			HOME_PHONE	
TRANSFER DATE		DOB	UNIT/CURRICULUM			WORK_PHONE	
PCS DATE		SPONSORED BY SSN					

VEHICLE

PLATE #	STATE	OWNER SSN	COLOR	MODEL	MAKE
YEAR	DECAL #	BODY	EXP YR	EXP MON	

TICKET

NUMBER	DATE	TIME	VIOLATION	ISSUED TO SSN	
DISPOSITION	PTS	JUDGE	DIS_DATE	INDEX	STATUS
LOCATION	CLASS	COURT_DATE		WRITTEN BY	
DESCRIPTION	REMARKS				

Candidate Keys = Underline

Foreign Keys = Bold Underline

Figure 4. Schema Mapping

#### D. TABLE DESIGN

After the schema has been refined, the tables can be built. A primary key is selected for each entity from among its candidate keys (if there is more than one candidate key, the designer should choose the primary key based on which field is most likely to be used in future transactions such as queries.). Field types and sizes are decided upon (this is based mostly on experience however some developer books might offer suggestions for this) and

then the tables are built in a manner specified by the DBMS. The COPS for Windows table layout for the person and vehicle tables is depicted in Tables 1 and 2. An 'A' in the value type column represents an Alphanumeric type. This type allows any keyboard character or number. A 'D' in the value type column represents a Date type. This type must be in a format specified by the DBMS it can allow a MM/DD/YY, DD MON YY format or one of several other variations.

VEHICLE TABLE			
COLUMN NAME	VALUE TYPE	LENGTH	KEY
LICENSE NUMBER	A	10	PARTIAL
LICENSE STATE	A	2	PARTIAL
DOD DECAL	A	6	
EXP MONTH	A	3	
EXP YEAR	A	4	
MAKE	A	10	
MODEL	A	10	
COLOR	A	6	
YEAR	A	4	
OWNER SSN	A	11	FOREIGN
BODY	A	1	

Table 1. Vehicle Table

PERSON TABLE			
COLUMN NAME	VALUE TYPE	LENGTH	KEY
SSN	A	11	PRIMARY
ADDRESS_CITY	A	17	
ADDRESS_STATE	A	2	
ADDRESS_STREET	A	30	
ADDRESS_ZIP	A	5	
BRANCH_SERVIC	A	6	
DOB	D		
LICENSE_NUM	A	11	
LICENSE_STATE	A	2	
DUTY STATION	A	10	
FACULTY_STAFF	A	2	
GRADE	A	4	
HOME_PHONE	A	10	
NAME_FNAME	A	10	
NAME_LNAME	A	15	
NAME_MI	A	1	
PCS_DATE	D		
RANK	A	3	
SMC	A	4	
SPONSORED_BY	A	11	FOREIGN
TRANSFER_DATE	D		
UNIT_CURRIC	A	3	
WORK_PHONE	A	10	

Table 2. Person Table



## **E. TABLE CONSTRAINTS**

Constraints are placed on fields within a table to insure that the data entered will conform to a specification. Although the field type is a constraint, it is generally too broad a type to ensure the needed consistency. Common constraints include mandatory fields, default values, minimum/maximum values, and "pictures" or a list of acceptable field entries.

### **1. Referential Integrity**

Referential integrity is a complex constraint. It is used to ensure data consistency and integrity among different tables that have a one-to-many relationship. An example would be the parent-child relationship between person and pet. Person is the parent, and pet is the child. It is a one to many relationship because a person may own zero or more pets, however a pet must be owned by exactly one person. Referential integrity constraints can prevent:

- A child record from being inserted without an existing parent record in place.
- Deleting a Parent record when it has associated child records.
- Mismatched key fields due to update anomalies by preventing or cascading parent-record key changes.

Referential integrity constraints were established for the following relationships:

- Person - Vehicle
- Person - Ticket
- Person - Pet
- Person - Weapon
- Person - Faculty Decal
- Person - Storage
- Person - User

## **2. Secondary Indexes**

The data tables are maintained in a sorted order according to their respective key fields. This ordering increases the speed of the operations that require looking through the table. These operations might include queries, joins, insertions, and deletions. These operations will only enjoy this speedup if the operation includes the key. A query for a person with the last name starting with the letter 'N' will take longer than a query for a person with an SSN starting with the number '3'. If a table designer knows that there will be a significant number of operations performed on other-than-key fields, then the designer can implement secondary indexes on these other fields to increase the efficiency of these operations. For the COPS for Windows application, secondary indexes were created for all of the child records listed in the above referential integrity constraints with the parent's key field, and the person table has a secondary index of the last name.

### **F. IMPLEMENTATION**

The last thing to be done before the tables can be set up is the determination of the field lengths. This is sometimes a best guess about how many characters will fill the field (e.g., 15 characters for a last name, 11 characters for an SSN). Some field types will not require a length such as a numeric or date field because these types are specified by the underlying database and the operating system. If exact information is not available as to the length of what is to be stored within a specific field, the designer must draw upon his own experience to estimate a field length. Since COPS for Windows is a second generation application, the field lengths and data types were chosen to maintain compatibility with the earlier data files.

### **G. DATA CONVERSION AND VALIDATION**

The existing data from the old version of COPS was used to populate the new data tables. In particular, three old files were converted: one each for Person, Vehicle, and Ticket information. The first step was to copy each of these files into a Paradox table format. Once the tables were copied. Scripts written using the Paradox programming language, ObjectPal (OPal), queries, and constraints were used to normalize different fields within each table.

During this process, there were primarily two different types of errors found, redundant data, and orphaned records. Redundant records are more than one record with the same key field. These errors were found and handled as described below. The only other error found was illegible data. When illegible records were found, they were discarded.

### **1. Ticket Table Conversion**

The ticket table required three fields, "violation date," "court date," and "disposition date," to be converted. This required a string conversion into dates by breaking the field apart then concatenating it with appropriate date separators and casting the whole thing as a date type ("120495", a six-character string, became 12/04/95 a date). There were more than 2,000 ticket records converted. Less than 5% of these records were discarded.

### **2. Vehicle Table Conversion**

The vehicle table needed the license plate field broken into two fields; state, and number, and the decal field needed to have the two character color code stripped from the decal number prefix (CA3BOW420 became State: CA, Number: 3BOW420, and NBXDT111 became Decal: XDT111). This script is shown in Figure 5. There were more than 9,000 vehicle records converted. Less than 2% of these records were discarded.

### **3. Person Table Conversion**

The person table was the most complicated conversion. It required breaking the name field into three separate fields for first name, last name, and middle initial, and breaking the address field into three separate fields for street, city and state. Although some of this was accomplished with OPal, because of field inconsistencies most of this conversion had to be done by hand. The name field had some consistency. It was usually formatted with last name first, followed by a comma, a space, then the first name and sometimes a middle initial that allowed a script to break this apart. However, there was no consistency in the address. Therefore the address field was converted manually. Whenever possible, queries were used to make information consistent (e.g., zip codes 93940 with a city starting with the letter "M" was changed to make every city "Monterey"). There were more than 6,500 person records converted. Less than 5% of these records were discarded.

```

method run(var eventInfo Event)
VAR
    SearchTC      tcursor
    before        string
    olddecal      string
ENDVAR

IF SearchTC.open(":cops:decal.db") THEN
    SearchTC.edit()
    SCAN SearchTC for SearchTC."mod" = false:
        message(SearchTc.recno())
        before=SearchTC."Lic_No"
        olddecal=SearchTc."Decal_NO"
        SearchTC."Decal_NO"=olddecal.substr(3,6)
        SearchTc."Lic_No" =before.substr(3,10)
        SearchTc."Lic_state" =before.substr(1,2)
        SearchTC."mod"=true
    ENDSCAN
    searchTC.endedit()
    searchTC.close()
ELSE
    beep()
ENDIF
endmethod

```

Figure 5. Sample Paradox Script

#### 4. Remove Redundant Data

Redundant data was removed by using the key integrity constraint implemented within the DBMS. This constraint simply states that the key must have a unique value for every instance of that entity type. For each table, a key was assigned. If there were any violations of this constraint, the DBMS kept the numerically first duplicate record it processed and discarded the rest. SSN was chosen as the key for the person table. License plate and license state together formed a composite key for vehicle. Since decal was a candidate key it was also assigned as a key value. The citation number was assigned as the key value for the

ticket table. After this, each table was viewed, and any entry that was for whatever reason still illegible was removed.

### **5. Remove Orphan Records**

Orphan records are those child records that have a foreign key that does not point to an existing parent record. To remove these orphans, referential integrity constraints were imposed with the person table as the parent, and the vehicle and ticket tables as the children. All orphan records were discarded.

## **H. INSTALLATION**

Paradox runtime version 5.0 was installed onto each individual PC. Populated data tables were placed into the shared network directory. The network administrators insured that all COPS users view the shared directory as the same drive letter, and they each have read, write, and creation permissions for that directory. IDAPI configuration included the settings for local share to be set to True, the network directory set to point to a network shared directory (the same directory for each user), and an Alias for "TABLES" set to the <data directory>. Forms and reports were "delivered" (this compiles all code included on the forms to ensure that end users cannot change them or modify the reports in any way), and, along with the queries, were loaded into the users working directory. By using the Windows file manager, an association for ".FDL" files is set for Paradox Runtime. An icon is created for the form WELCOME.FDL, and the PC's CONFIG.SYS file is modified to load SHARE.EXE.



## IV. SYSTEM VALIDATION

### A. INTRODUCTION

The testing method chosen for this application was functional black-box testing. This means that the functionality described in the requirement's document (Appendix B), will be tested, however there will be no attempts to verify or validate the underlying DBMS.

The first step in development of the test plan is to list all the testable requirements. This list is found in Appendix C and tends to focus in three areas: how the data is entered, what it looks like when it is retrieved, and which users can do what functions. Next, the developer must classify these requirements by how they can be tested. These categories are then grouped together so that each test case will cover as many requirements as possible. This is called aggregation and is done to reduce testing cost (cost measured in both time and money). After aggregation, the developer begins making test cases. These cases are a detailed listing of exactly what is to be done, what data is used, what steps are followed and what the expected results will be.

### B. REQUIREMENTS CLASSIFICATION

Depending on how the testing is performed, requirements are normally classified into four different categories: Non-Testable, Inspection, Analysis, and Execution. Non-Testable includes items such as vocabulary definitions, and user actions. Inspection includes looking at the actual code for items such as language usage and documentation. Analysis includes looking at the actual code for items such as variable usage and commenting. Execution includes running the application (or testable modules) and looking for items such as calculation results and value transformations.

For this testing process all of the requirements are classified as Execution and must be tested during the actual running of the application except the following requirements. Requirements 25, 27, 32, 33, 35, 36, and 37 are classified as inspection. These requirements are concerned with field format (required field or acceptable values) or referential integrity. Although they could be tested during execution, viewing the underlying table structure to

insure the appropriate constraints are in place is much easier. Requirement 40 (networkable with up to 75 simultaneous users) is non-testable due to a lack of available resources.

### **C. AGGREGATION**

Aggregation is the combination of the classified requirements that can all be tested during one test run. The intent of aggregation is to reduce the overall number of test cases needed to be run and thus reduce testing time and cost. There were eight aggregate test cases, one for each class of user and one for miscellaneous requirements. The remainder of this chapter describes a broad overview of some tests run and the results.

### **D. TEST CASE OVERVIEW**

The test cases included: repeatedly logging in as different user classes to verify function availability; issuing an unknown person a parking ticket with only the vehicle decal for information; issuing a parking ticket to an unregistered vehicle (application must generate SSN); registering a user for the first time (with and without the application generating an SSN); creating a Docket, Suspension Letter and Letter of Acknowledgment; and using the Browse function to search for a user and view all related information for that user. In short, every button on every form was pushed at least once to insure the results were as expected. For data entry, at least two test cases were run for each aggregate, one for a known person (issuing me a ticket or registering my own vehicle . . . ), and one for a fictitious person (John Doe with a system generated SSN). After the initial interface was created that allowed 75% functionality for the Registration Clerk and the Traffic Administrator, a copy of the application was installed on each of their PC's for "Beta Testing."

### **E. SUMMARY OF RESULTS**

During the preliminary testing less than ten bugs were found. These bugs minor and included errors of omission (the Monterey tag number was not included as a field in the pet table), and interface errors (print on reports misaligned). The "Beta Testing" however was very thorough as the application was put through its paces by the actual users. Most of the complaints from the users turned out to be from lack of understanding of how the system worked, or using the system in a way that it was not designed. One of the most memorable



bugs related to a ticket being issued to a person driving a vehicle registered to someone else. The application would search the data to find the owner of the vehicle, then issue the ticket to that person. When the ticket was reconciled with the person it was actually issued to, if that person's SSN was placed into the key field, it would be cascaded down to all of its child records corrupting the data. This led to a new requirement that tickets shall be able to be deleted or re-associated to a different person.



## **V. CONCLUSIONS**

### **A. RESULTS**

This thesis stepped through the complete development cycle for a small-scale information system much like one that a small business or organization might use. By using the components of a DBMS to develop the interface, reports and tables, this programming-in-the-small becomes much more economical for a smaller, less experienced development team than it has previously been. For small businesses or organizations this is important for becoming more efficient both in customer service and office management. For developers, this is a good case-study for the development of an SSIS, and can serve as a guideline for creating their own development process. The NPS Police Department has found that after the initial growing pains of upgrading their software, they have already become more efficient at their tasks of data entry and retrieval. Tickets are entered in a more timely fashion because the shared database eases information searches. Reports that used to take hours are now done in minutes and with greater precision. The information is available at all levels, from the desk clerk to the police-chief. Everyone has the information right where it is needed, at their fingertips.

### **B. RECOMMENDATIONS**

Although the power of the personal computer (PC) is growing, and with it the DBMS applications, there is still room for improvement. More research should go into the development of SSIS applications with fourth and fifth generation languages and tools such as Visual Basic by Microsoft, or Delphi by Borland. A recommendation specific to NPS is the reactivation of a DBMS development team within the Information Management Section. This team can create applications such as COPS for Windows for other departments within NPS.

### **C. FUTURE WORK**

Future work in this area includes a continuation of the development process of COPS for Windows, in particular, the maintenance cycle. After the end users work with the new

application for a while, shortcomings and new features might be identified. This cycle would primarily work on improving and upgrading the application. Some of these upgrades might include more reports such as a form letter to students informing them of expired vehicle registration, or the implementation of the Scantron input device for the Registration Clerk. Further development with other NPS departments is also a possibility. This might include the integration of more data and other user classes if other departments decide to participate in the data sharing (e.g. add the S-2 or the Registrar).

Future work could also include the development of tools to help with this methodology. These might include a drawing tool to create E-R diagrams and then transform them into database tables or a testing tool that could analyze an E-R diagram to determine specific test cases.

## APPENDIX A. USERS MANUAL

*take over hyphen  
"mouse use"*

### A. INTRODUCTION

This is the user's manual for the COPS for Windows application. It is written under the assumption that users are familiar with basic Windows commands and mouse use.

### B. INSTALLATION

This section steps through the exact installation process required to install COPS for Windows on a PC at NPS.

- Installers must coordinate with the IMS department to have access to the COPS directory and runtime subdirectory.
- Install Paradox Runtime version 5.0 from the network runtime directory onto the user's PC.
- Use File Manager to copy all the COPS forms from the COPS for Windows form subdirectory into the runtime working directory.
- Enter the IDAPI configuration utility found in the runtime group on the windows desktop to make the following changes: Set an Alias for Tables to the location of the COPS shared data files (R:\COPS4WIN). Set local share to True. Set network directory to the location of the file PDOXUSRS.NET (R:\).
- Use the File Manager to create an association for .FDL files to the Runtime executable.
- Use the Program Manager to create an ICON for the WELCOME.FDL file and rename it COPS for Windows.
- Use an editor such as Notepad or the DOS Editor to modify the CONFIG.SYS file. (Add line: Device = C:\DOS\SHARE.EXE)
- Reboot computer.

### C. LOGGING IN

The first thing every user must do to use this application is to log in. After clicking on the COPS for Windows ICON, the “welcome” screen will appear on the screen. The user should either click anywhere on the welcome screen or hit the enter key. This will bring up the login screen shown in Figure A-1.

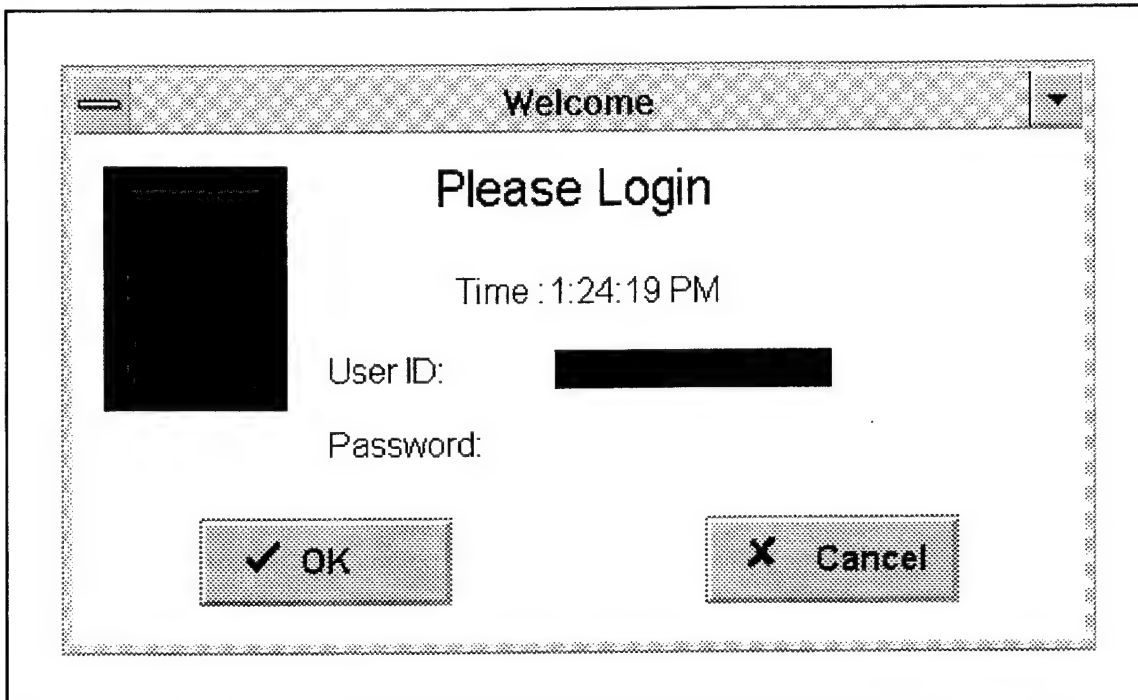


Figure A-1. Login Screen

The user should then enter his user id and password into the appropriate field (both fields are case sensitive). After these fields are filled (the password field will not echo any typed characters) press the OK button on the bottom left of the screen. This application comes with a default system administrator account that can be used for the initial setup of the user accounts. The user id is “sysop”, the password is “Password.” This account should be deleted as soon as another system administrator account has been established.

### D. MAIN MENU

After logging in, the user will see one of seven different main menus dependent upon what user class he is assigned. Each of these main menus gives the user all the different

functional options he or she has access to (Browse, Reports, Registration, Tickets . . . ). The main menu for the Traffic Administrator class is shown in Figure A-2.

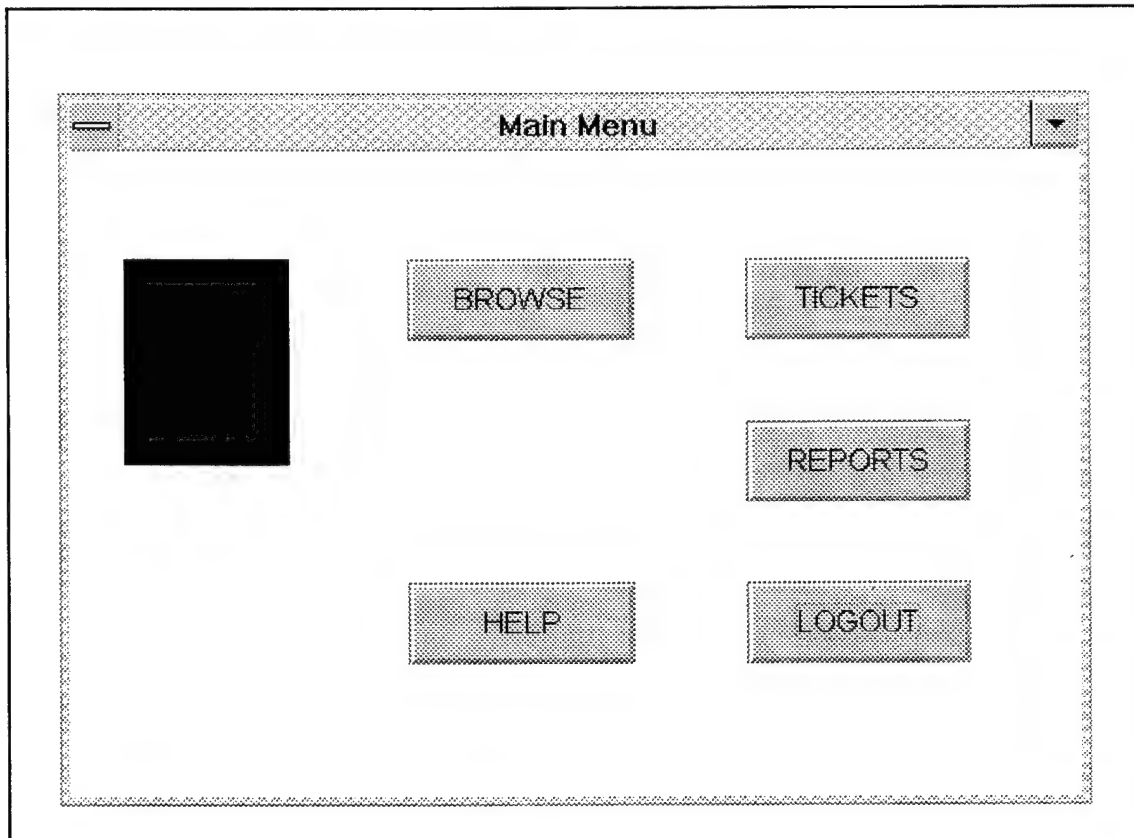


Figure A-2. Traffic Administrator Main Menu

Clicking the buttons with the left mouse button will bring the user to another set of choices or to a form that needs to be filled. All the buttons are labeled with a short (one word) description of what that button does. All main menus look similar to the Traffic Administrator's menu shown previously.

#### **E. DATA ENTRY**

Data entry is performed similarly for each different type of user. The menu system is navigated until the desired form is reached (all forms are modeled after existing paper documents for familiarity and ease of data entry). Some decisions may have to be made along the way or the application may ask for information such as a person's SSN or a citation

number. When something is needed, special dialog boxes pop up and ask specifically for the desired information.

## **F. TICKET**

The ticket function is available to the Traffic Administrator and Chief class users. The ticket button is used to enter newly issued citations, modify existing citations after reconciliation, and to enter personnel on the various maintained lists (DUI, Probation, or Suspension).

### **1. New**

This button is pressed to enter a citation that has never been entered into the database before. This button has three mutually exclusive options attached to it: parking ticket, moving ticket and new record. These three options coincide with what information is available when the ticket is written. If the ticket is for a moving violation, it was issued to a person therefore the application prompts the user for an SSN. If the ticket is a parking violation, the only information available is the registration information, therefore the application prompts the user for decal information. If no information is available to associate the vehicle with a registered person, new record is selected.

### **2. Modify**

This button is pressed to edit a ticket that has been previously entered such as during traffic court. When the button is pressed, the user is prompted to enter a citation number. The application searches the database for the entered citation number. If it is found, the main ticket screen will appear with the person's information showing to whom the ticket was issued.

### **3. Suspension**

Pressing this button will bring up a new dialog box. This dialog box has three buttons on it: letter, new, and done. Pressing the letter button will bring up a dialog box that prompts the user for the SSN of the person to whom the letter will be issued. A form letter then appears with all the heading and addressee information filled in. The user can then edit the three paragraphs of the letter (user must push the next page button to get to paragraphs two



and three) and the copy to field. When the user is done editing these fields, the user may push the acknowledge button. This will bring the user to the letter of acknowledgment page. This page is edited just as the previous page was. The user can then press the print button to print the letter. If a suspension letter has already been generated or is not necessary, the user can enter a person onto the suspension list by pressing the new button. This will bring up a dialog box prompting for an SSN, start date, end date, and suspension type.

#### **4. Probation**

This button works just like the suspension button except the letter. The probation report does not generate a letter. Pressing this button brings up a dialog box that prompts the user for an SSN, start date, end date, and type. Pressing the done button commits the record to the database.

#### **5. DUI**

This button is exactly like the probation button but users will be placed on the DUI list and there is no ending date to be filled. Pressing the done button commits the record to the database.

### **G. REPORTS**

The reports function is available to the Chief, Traffic Administrator, and Desk Clerk class of user from their main menu. Pressing this button brings up the report menu that allows access to the various generated reports.

#### **1. Docket**

By pressing the docket button, a Docket will be printed. The Docket will include all citations with a court date greater than <today's date - 1> that has an open status. These parameters allow the printing of the docket on the day of court. As tickets get closed during court, they will be removed from the Docket if it is reprinted.

#### **2. Statistics**

By pressing this button, a new screen will appear. This screen contains two fields, one for the month of the report (Jan. = 1, Dec. = 12), and the other for the fiscal year that the report is in (i.e., October of 1994 is month 11 of fiscal year 1995). If the statistics are already

generated for this month, a question box will prompt for recalculating the statistics. This prompt, allows for recalculation if the user feels enough tickets were entered after the report was already generated (all subsequent reports would also need to be recalculated to insure correctness of the FY totals, the monthly average, and the monthly difference). This function is time consuming (approximately 1 minute for 2,500 records) because it must search through every ticket for every query.

### **3. Suspension**

By pressing the suspension button, a Suspension Report is printed. Each time this button is pressed, the application will perform maintenance on the underlying table. All persons whose suspension's ending date is less than <today's date> will be removed from the list. The application generates a report of those names removed from suspension.

### **4. Probation**

By pressing the probation button, a Probation Report is printed. Each time this button is pressed, the application will perform maintenance on the underlying table. All persons whose probation's ending date is less than <today's date> will be removed from the list. The application will generate a report of those names removed from probation.

## **H. BROWSE**

The browse function is available to all user classes. It allows the user to view those persons registered and all related registration information (tickets, vehicles, weapons . . . ).

Pressing one of the large buttons will bring the user to another screen with the related data indicated on the button (the pet button will display that person's registered pets . . . ). The database records are stored in ascending order by SSN. The smaller buttons are used for navigating through these records. The arrows pointing to the right move down the database records (from the current record to the end of the data). The single arrow moves one record. The double arrow moves 20 records, and the arrow with the bar moves to the last record in the database. The arrows pointing to the left do the same functions as their counterparts but in the opposite direction. The magnifying glass is a locate function that allows the user to perform searches based on a single field. The magnifying glass with the dots is a locate next

function. Pressing this button will find the next occurrence of the parameters specified in the locate function.

## **I. DESK JOURNAL**

The desk journal function is available to the Desk Clerks and the Chiefs. The Chiefs have the added capability that from within the view mode of the desk journal, they can edit and validate the Journal.

### **1. Open**

When the user presses this button, a dialog box will prompt the user for a date for the desk journal. Only one Journal per date may be created. If the desk journal for that date already exists and is not validated, the user may append entries to the journal. If the desk journal does not exist, a new desk journal will be created. A dialog box confirms that a journal was opened.

### **2. Close**

Pressing this button will close the current desk journal. A dialog box prompts the user for confirmation and another will confirm that the Journal was closed.

### **3. View**

Pressing this button allows the user to view the currently open Desk Journal. From this screen, the user has the option of printing the form. If the user class is a chief, then two additional buttons will appear on this screen, validate and edit. The edit button will allow the user to edit any of the fields on the desk journal if it has not been previously validated. Pressing the validate button, validates the Desk Journal. Validating the desk journal prevents any further editing.

### **4. New**

Pressing this button will allow the user to make a new entry into the currently open Desk Journal (if it has not been validated). The application will automatically fill the entry number and the entry time. Users must fill the description and the information block. Once the user presses the done button, the entry will be committed to the journal and only a chief can edit it.

## **J.     LOGGING OUT**

In most cases, starting a new action (registering a new person on base, entering a citation that was just issued) is started by pushing the respective new button. To edit an existing record the modify button should be used. In all cases, when an action is complete, the done button should be pushed. Pushing this button will eventually lead the user back to his own main menu. From here, the user can log out of the application by pressing the logout button on each main menu. Because this application runs on top of the runtime shell, the user will also need to exit this program. This is done just as in any other Windows application; use the mouse to select the File menu option then choose Exit.

## **APPENDIX B. REQUIREMENTS DOCUMENT**

### **A. INTRODUCTION**

The Naval Postgraduate School Police Department needs a small scale information system that allows multi-user access (possibly from remote sites), to vehicle, bicycle, RV, and pet registration data, ticket information, daily journals, and incident reports. It should be easy to use (users should be fully functional with an hour of training), provide security for the data (mandatory access controls for the application interface), and allow for the future growth of the department (allows multiple users from remote locations). Its overall objective is to improve productivity and efficiency by automating the tasks and records keeping that are currently done manually.

### **B. SYSTEM MODEL**

The COPS for Windows system is designed to be run as part of a network with multiple users sharing the same data files, or as a stand alone version with a single user and a single data set. The NPS Police will be using it over a network with several different users as depicted in Figure B-1. All these users need access to the same information but are found in different areas of the building. As the department expands to take on more areas of responsibility, such as Presidio of Monterey (POM) and the POM annex, the users could even be in different parts of the city. This access will be provided through a Banyan Vines network and a shared file service. The primary users of the system are the Traffic Administrator, the Registration Clerk, the Desk Clerk(s), Casual Users (view only), Chiefs, Armorers, and the System Administrator.

The Traffic Administrator's primary functions include entering ticket information, generating suspension letters, generating and distributing suspension reports, generating the "docket" (a listing of who goes to court and when), generating ticket analysis reports, and file maintenance of all the above-mentioned reports and information. All data entry and interaction is done manually based on a keyboard and mouse.

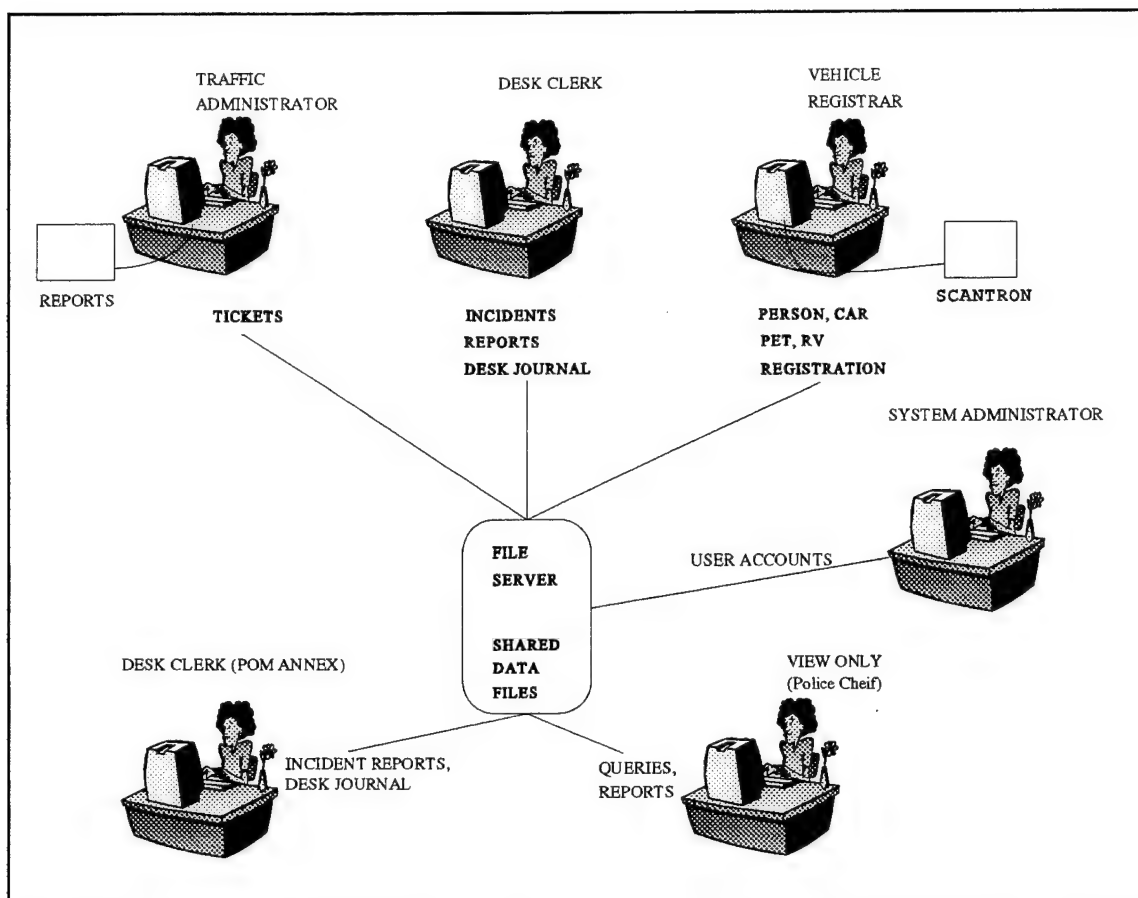


Figure B-1. System Overview

The Vehicle Registration Clerk is primarily responsible for populating the database with information about people, vehicle registration, pet registration, and RV registration data. The Registration Clerk must ensure that all these files are maintained which includes editing the information when it changes and archiving old records. The Registration Clerk does not generate any reports. Data entry can be done manually based on a keyboard and mouse or automated with a Scantron Model 8200 Optical Mark Reader.

The Desk Clerk's primary function includes the maintenance of the Daily Journal, and completing Incident Reports. The Daily Journal is a log of events similar to a diary. The

Incident Reports are a detailed accounting of certain events such as a misdemeanor or a complaint about somebody's dog.

Casual users are the users that have a primary function other than working with the database. For these users, the database is a tool with which they can do their related primary duties. These users have access to reports and a browsing/query function but are not able to edit the data.

Chiefs have the most access. They have all the same functional needs as the Casual User, and the Traffic Administrator. They also are allowed to verify the Desk Journal and the I/C Reports. Verification is a one time chance to edit the reports then commit them to the Database for storage or viewing.

Armorers are primarily interested in weapon storage information. They maintain a record of who owns which weapon. The Armorers do not generate any reports. However they do need to be able to view their database of weapons and who owns them.

The System Administrator's primary responsibility is the maintenance of users of the application. The system administrator must set up and maintain user accounts. Although a system administrator may delete user accounts, he or she can not delete his or her own account thus ensuring that the application will always have at least one administrator. The Administrator has no other access.

### **C. SYSTEM EVOLUTION**

The system evolved from a DOS-based environment. Each user had their own data sets and would try to maintain consistency by exchanging the files that they each updated. If a user needed information that he or she did not have, they would ask someone else or try to find it manually.

Installation of COPs for Windows will be phased over two months. Existing registration and ticket data files will be used to populate the database. Test versions of the application will then be installed on the Registrations Clerk's, Traffic Administrator's, and the Police Chief's PCs and used for daily business and testing. Finally the application will be installed on the remaining PC's. The users will be given instruction at the time of installation

on how to use the application. Users should continue to mirror their transactions by hand until they have a high degree of confidence in the application.

## **D. FUNCTIONAL REQUIREMENTS SPECIFICATION**

### **1. Reports**

- 1.1. The Application shall provide a mechanism to generate the following reports in the format that matches current paper reports:
  - 1.1.1. Suspension Letter (see Figure B-2)
    - 1.1.1.1. Letter of Acknowledgment (see Figure B-3)
  - 1.1.2. Docket (see Figure B-4)
  - 1.1.3. Incident / Complaint Report (I/C) (see Figures B-5, B-6, B-7)
  - 1.1.4. Desk Journal (see Figure B-8)
  - 1.1.5. Suspension Reports (see Figure B-9)
  - 1.1.6. Ticket Analysis (see Figure B-10)
  - 1.1.7. Citation Listing (see Figure B-11)
  - 1.1.8. Probation Report (see Figure B-12)
  - 1.1.9. DUI Listing (see Figure B-13)
- 1.2. The reports shall be displayed on screen by default with an option to send it to the printer.

#### **Rationale:**

Maintaining the information on a computer does not aid anyone if the users can not view or manipulate the data, or have the application do work based on the data. These are the reports that are manually generated regularly. By having the computer generate these reports, a significant amount of labor and time will be saved.

### **2. Forms**

- 2.1. All forms used for data entry shall conform to existing documents.
  - 2.1.1. Vehicle Registration (See Figure B-14)
  - 2.1.2. Pet Registration (See Figure B-15)



- 2.1.3. RV Storage (See Figure B-16)
- 2.1.4. Ticket Data (see Figure B-17)
- 2.1.5. Desk Journal (see Figure B-8)
- 2.1.6. Incident / Complaint Report (see Figures B-5, B-6, B-7)
- 2.2. Incident / Complaint Report information will be broken into several smaller forms so they may be viewed legibly one section at a time on a VGA monitor.
- 2.3. Ticket information will be broken into several forms so they may be viewed legibly one section at a time on a VGA monitor.

**Rationale:**

Forms are used to view the data while entering information. These forms should closely mirror the actual forms used on paper. This duplication will speed up the processing time of the data input by saving the clerks the frustration of trying to figure out which bit of data goes in which box. If the screen looks identical to the paper they are holding in their hands, the puzzle should fit together nicely. If however the form is too small to be legible during input, the form should follow the design of the paper document but is not held to actual size requirements.

**3. Interface**

- 3.1. The application shall have a graphical user interface. The interface should be menu driven, tolerant of mistakes, and easy to use<sup>1</sup>.
  - 3.1.1. Clearly defined menu options should be made available based on the type of user and that user's required functionality.
    - 3.1.1.1. The Traffic Administrator needs to have access to Ticket entry, Browse, and Reports functions.

---

<sup>1</sup>Ease of use is a fairly subjective term. It can however be quantized by empirical information such as hours of training required for a novice user to become proficient with the application and measurable productivity gains or losses.

- 3.1.1.2. If a ticket is being issued to a person that is not currently in the database, the Traffic Administrator needs access to registration data to enter that information.
- 3.1.1.3. The Registration Clerk needs access to the registration and Browse functions.
- 3.1.1.4. The Desk Clerk needs access to the I/C Reports, Desk Journal, Browse, and Reports functions.
- 3.1.1.5. Casual Users need access to the Browse functions.
- 3.1.1.6. The System Administrator needs access to the System Administration Functions.
- 3.1.1.7. The Traffic Administrator will be responsible for file archival and restoration, and maintenance of the validation tables.
- 3.1.2. The application will provide validation tables, referential integrity lookups, and formatting information for field entry.
- 3.2. The user shall be able to view related information when browsing through records.
  - 3.2.1. While looking at a record, the user shall be able to move to another window with pertinent ancillary information. (e.g. can jump to a vehicle record while viewing personal information.)
- 3.3. The Traffic Administrator shall be able to enter ticket information based on a SSN, or a DOD Decal number.

**Rationale:**

A Graphical User Interface, or GUI as it is more commonly known, tends to be easier to understand and use both for the novice and experienced user. The users of the proposed system have limited computer skills. As such, it is needed that the system be easy to learn, understand, and operate.

#### **4. Security**

- 4.1. The system shall provide mandatory access controls.
  - 4.1.1. Each user shall have a password and a security level. The password will allow the user access to the application. The security level will allow the users different levels of access to the information.
  - 4.1.2. Passwords shall be between seven to ten case-sensitive alphabetic characters in length and should be changed every four weeks.

##### **Rationale:**

To maintain data integrity, only certain users can be allowed to modify the maintained data. Since much of the data is protected by the privacy act, only those with a need to view it can be given access.

#### **5. Scantron**

- 5.1. The system shall interface with a Scantron model 8200 optical mark reader to enter person and/or vehicle registration data (as defined in the database requirements section).

##### **Rationale:**

This is another means of entering information. By using this, the data should be less prone to errors caused by data entry.

#### **6. Queries**

- 6.1. Users shall be able to perform complex queries that include AND and OR conditions, partial identifiers (license plate ..324.), and involve multiple table joins.

##### **Rationale:**

Information must be easy to extract without having complete knowledge of the individual. For example, the user can query for all white Jeep Cherokee vehicles, or owners of a black dog.

## **7. Validation Tables**

- 7.1. The software shall maintain Validation tables to ensure consistency of data entry on specified fields.
  - 7.1.1. Specified fields are Ticket Disposition (open and closed index), State Abbreviations, and Traffic citation numbers.
  - 7.1.2. These validation tables shall be maintainable by the Traffic Administrator, and the Vehicle Registrar.

### **Rationale:**

The use of validation tables for data entry will help ensure data consistency. For example, CA will always be used for the state abbreviation for California, not Ca. or cA. or ca.

## **8. Goals**

- G.1. The system should allow cascade archival of personnel records and all child records when that person checks out on PCS orders.
- G.2. Incident / Complaint Reports should be archived after they become two years old.
- G.3. Integrity Constraints
  - G.3.1. A ticket must be issued to a person in the Database.
  - G.3.2. A vehicle must belong to a person in the Database.
  - G.3.3. A ticket must have a disposition from either the open or closed Index validation table.
  - G.3.4. A person must have an SSN to be entered into the database.
    - G.3.4.1. The application shall allow a standard form SSN of nine digits, will generate an SSN of the form XXX-YY-#### where XXX are literal characters, YY stands for the current year, and #### will be an application generated serialized number, or will allow a foreign student ID number of the form CCC-YY-

Q### where CCC is a character abbreviation for their home country, YY is the year they report, Q is the quarter they report, and ### is a serial digit provided by the registrar.

## **9. Constraints**

C.1. Desk journal entries once committed are not modifiable by anyone.

## **10. Assumptions**

A.1 Backup and restoration policies will be implemented by the LAN administrator and are not part of this application.

A.2 Security of the actual data files will be dependent upon the network security provided by the Banyan Vines networking software.

## **E. NON-FUNCTIONAL REQUIREMENTS SPECIFICATION**

### **1. Hardware**

The application shall be able to run on existing hardware. This includes a minimum of Intel 386 compatible PC's with four Megabytes of RAM running Microsoft Windows ver. 3.1. These are the minimum requirements for running Paradox. A better system (8 Megabytes of RAM and an Intel 486 CPU) would improve performance of the application and is recommended.

### **2. Software**

The choice of the development DBMS is based almost completely upon economics. Since the School already has a site license available for all Borland Products, the application will be written in Paradox ver. 5 and delivered with Paradox Runtime ver 5.

### **3. Documentation**

The only required documentation will be a User's Manual and the accompanying thesis. The User's Manual shall include all necessary information for a user with average computer literacy skills to install and operate the application.

#### 4. Training

One hour of training shall be set aside per user.

#### 5. Installation

Installation of the application on the workstations is required. Population of the new data tables from existing registration data is also required.

#### 6. Maintenance

Maintenance of the application is the responsibility of the developer until graduation (Sept. '95) and thereafter Naval Postgraduate School Police department. The Computer Science department is in no way obligated for the maintenance of the application currently under development.

### F. DATABASE REQUIREMENTS

These are the attributes associated with each entity.

#### PERSON

SSN  
NAME  
    LAST  
    FIRST  
    MIDDLE  
ADDRESS  
    STREET  
    CITY  
    STATE  
    ZIP  
DRIVERS LICENSE  
    STATE  
    NUMBER  
RANK  
GRADE  
CURRIC/STAFF CODE  
FACULTY CODE  
WORK PHONE  
BRANCH OF SERVICE  
SMC #  
DUTY STATION  
HOME PHONE

TRANSFER DATE  
UNIT/CURRICULUM  
DATE OF BIRTH

#### TICKET

NUMBER  
DATE  
TIME  
LOCATION  
DISPOSITION  
    DATE  
    DESCRIPTION  
    JUDGE  
    POINTS  
COURT DATE  
INDEX  
STATUS  
WRITTEN BY  
REMARKS  
DESCRIPTION  
VIOLATION(S)

VICTIM/SUSPECT

SSN  
POB  
SEX  
RACE  
HEIGHT  
WEIGHT  
HAIR  
EYE COLOR  
ID MARKS

PET

NAME  
BREED  
TYPE  
SEX  
COLOR  
AGE  
WEIGHT  
NPS TAG #  
MONTEREY TAG #

VEHICLE

PLATES  
STATE  
NUMBER  
MAKE  
TYPE  
COLOR  
YEAR  
DECAL NUMBER  
EXPIRES  
MONTH  
YEAR

DESK JOURNAL

ENTRY #  
DATE  
TIME  
INCIDENT  
ACTION TAKEN

STORAGE

TYPE  
PLATE  
NUMBER  
STATE  
NPS DECAL #  
STORAGE SPACE

I/C REPORT

FROM  
TO  
VIA  
CCN  
SUBMITTED  
RETURN BY  
WHEN/HOW REC  
DATE  
TIME  
HOW  
INVOLVEMENT  
ASSUMED BY NIS  
DATE  
TIME  
RECEIVED BY  
NAME  
RANK  
JOB  
TYPE OF INCIDENT  
RELATED PERSONS  
DETAILS  
ENCLOSURES  
EVIDENCE  
REFERRED TO  
DISTRIBUTION  
REPORTING OFFICIAL  
NAME  
RANK  
TITLE  
SIG  
APPROVING OFFICIAL  
NAME

RANK  
TITLE  
SIGNATURE  
REPORT OF ACTION  
FROM  
TO  
VIA  
SUBJECT  
ACTION TAKEN  
DATE COMPLETE  
DETAILS  
NAME  
TITLE  
SIGNATURE

SUSPENSION  
START DATE  
END DATE  
TYPE

DUI  
EFFECTIVE DATE

USER  
PASSWORD  
PASSWORD DATE  
USER ID  
LOGIN DATE

WEAPON  
SERIAL NUMBER  
MAKE  
MODEL  
DESCRIPTION

FACULTY DECAL  
NUMBER  
EXPIRATION DATE  
MONTH  
YEAR

PROBATION  
START DATE  
END DATE  
TYPE



5560

NPS(44)

<Date>

MEMORANDUM

From: <Name>, Security Officer, Code 44

To: <Name, SSN, Address, Code ##>

Subj: TEMPORARY SUSPENSION OF DRIVING PRIVILEGES

Ref: (a) NAVPGSCOLINST 5560.5

(b) OPNAVINST 11200.5C

Encl: (1) Letter of Acknowledgement of Temporary Suspension of Driving Privileges

1. <Paragraph 1>

2. <Paragraph 2>

3. <Paragraph 3>

<Name>

Figure B-2. Suspension Letter

5560

NPS (44)

MEMORANDUM

From: <Name, SSN, Address/Code ##>

To: <Name>, Security Officer, Code 44

Via: Dean of Faculty, Code 07

Subj: LETTER OF ACKNOWLEDGMENT OF TEMPORARY SUSPENSION OF  
DRIVING PRIVILEGES

Ref: (a) Naval Postgraduate School ltr 5560 NPS (44) dtd 10 Mar 95

1. I hereby acknowledge receipt of reference (a). I understand all the provisions and that failure to comply with them would be a violation of the Naval Postgraduate School Traffic Regulation (5560.5 dated 1 Feb. 1994). I understand that my base driving privileges may be revoked for up to one year and 12 points will automatically be assessed to my base driving record if I fail to comply with all provisions of the Naval Postgraduate School Instruction.

---

(SIGNATURE)

---

(DATE)

Figure B-3. Letter of Acknowledgement



[illegible]

54

DEPARTMENT OF THE NAVY INCIDENT/COMPLAINT REPORT (Continued)

17. DETAILS OF INCIDENT (Who, What, When, Where, How, Why? Attach Relevant Statements)

Figure B-6. Incident / Complaint Report Page Two

<b>DEPARTMENT OF THE NAVY INCIDENT/COMPLAINT REPORT</b> <i>(continued)</i>			
18. ENCLOSURES (Statements and receipts)		19. EVIDENCE (List and describe)	
20. REFERRED TO <input type="checkbox"/> Patrol <input type="checkbox"/> Investigation <input type="checkbox"/> NIS <input type="checkbox"/> File <input type="checkbox"/> Other Agency (specify)		21. DISTRIBUTION ORIG: _____ COPY 1: NISHQ via NISRA _____ COPY 2: _____	
22. REPORTING OFFICIAL TYPED NAME RANK/TITLE & SIGNATURE		23. APPROVING OFFICIAL TYPED NAME, RANK/TITLE & SIGNATURE	
24. REPORT OF ACTION TAKEN (To be completed by the addressee when so indicated in block 7. <u>Return one copy</u> to originator to meet suspense date indicated in block 6.)			
a. FROM		b. DATE	
c. TO			
d. VIA			
e. SUBJECT		f. RANK	g. SSN
h. ACTION TAKEN	<input type="checkbox"/> ADMINISTRATIVE	<input type="checkbox"/> NON-JUDICIAL	<input type="checkbox"/> JUDICIAL
j. DATE ACTION COMPLETED			
j. DETAILS (Specify type administrative action taken, non-judicial punishment imposed, or judicial results, as applicable)          <div style="text-align: center; font-size: small;">(For multiple subjects, use additional page(s) to reflect action taken.)</div>			
k. TYPED NAME AND TITLE		l. SIGNATURE	

Figure B-7. Incident / Complaint Report Page Three



SUSPENSION REPORT				
LNAME	FNAME	DATE START	DATE OVER	TYPE

Figure B-9. Suspension Report



## TRAFFIC VIOLATION STATISTICS

<MONTH YEAR>

	Current	Total (FY)	AVG	INC	DEC
Base Moving					
Base Parking					
La Mesa Moving					
La Mesa Parking					
POM/DLI Moving					
POM/DLI Parking					
POM Annex Moving					
POM Annex Parking					
Other Moving					
Other Parking					
Total:					

<Name>

Figure B-10. Traffic Statistics

CITATION LISTING					
LNAME	FNAME	CITATION #	DATE	VIOLATION	DISPOSITION

Figure B-11. Citation Listing

PROBATION REPORT				
LNAME	FNAME	DATE START	DATE OVER	TYPE

Figure B-12. Probation Report

DUI REPORT			
LNAME	FNAME	SSN	DATE

Figure B-13. DUI Report

Transfer Date: \_\_\_\_\_

NAVAL POSTGRADUATE SCHOOL BASE POLICE STUDENT / STAFF REGISTRATION					
NAME (Last, First, Mid Init / Rank / SSN)					
Office Activity / Telephone					
Residence Address Telephone					
Branch of Service	Decal Color	DOB	Drivers License	State of Issue	ID. Card Number
Vehicle (Year, Make, Model, Color)			State	License Plate Number	Decal Number / Exp Date (Month/Year)

Figure B-14. Vehicle Registration Form

PET REGISTRATION		
Name of Pet	Breed	
Dog / Cat	Sex	Color
NPS Tag Number	Monterey Tag Number	
Remarks		

Figure B-15. Pet Registration Form

RECREATION VEHICLE STORAGE			
RV Type (Boat, Trailer, ETC...)	License Plate	State	NPS Decal Number

Figure B-16. Recreation Vehicle Storage Form

5. ORGANIZATION OR ADDRESS					
6. DRIVER LICENSE NUMBER			7. ISSUING AUTHORITY (State or Military)		
8. MAKE OR TYPE OF VEHICLE		9. STATE LICENSE OR REGIS NO.		10. INSTL TAG NO.	
11. DATE (Day-month-year)		12. TIME		13. LOCATION	
V I O L A T I O N	<input checked="" type="checkbox"/> SPEED OVER LIMIT ( <i>mph in a mph zone</i> )	<input checked="" type="checkbox"/> 5 - 10 MPH	<input checked="" type="checkbox"/> 11 - 15 MPH	<input checked="" type="checkbox"/> OVER 15 MPH	
	IMPROPER LEFT TURN →	NO SIGNAL	CUT CORNER	FROM WRONG LANE	
	IMPROPER RIGHT TURN →	NO SIGNAL	INTO WRONG LANE	FROM WRONG LANE	
	DISOBEYED TFC SIGNAL (when light turned red) →	PAST MIDDLE INTERSECTION	MIDDLE OF INTERSECTION	HAD NOT REACHED INTERSECTION	
	DISOBEYED STOP SIGN →	STOPPED WRONG PLACE	FAILED TO STOP	ROLLED / SPED THROUGH	
	IMPROPER PASSING AND LANE USAGE →	AT INTERSECTION	CUT IN	WRONG SIDE OF PAVEMENT	
		BETWEEN TFC	ON RIGHT	ON HILL	
		LANE STRADDLING	WRONG LANE	ON CURVE	
	FOL. TOO CLOSELY	OTHER VIOLATIONS (Describe)			
	FAILURE TO YIELD				
PARKING		OVERTIME	DOUBLE PARKING		
		PROHIBITED AREA	OTHER (Describe in Remarks)		
C O N D I T I O N S  T H A T  I N C R E A S E D  S E R I O U S N E S S  O F  V I O L A T I O N	SLIPPERY PAVEMENT	RAIN	AREA	TRAFFIC ACCIDENT TYPE OF ACCIDENT:	
		SNOW			BUSINESS
		ICE			INDUSTRIAL
	DARKNESS	NIGHT	RURAL	PD	PI
		FOG	SCHOOL	FATAL	
		SNOW	RESIDENTIAL	PEDESTRIAN	
	OTHER TRAFFIC PRESENT	CROSS	HIGHWAY TYPE	VEHICLE	
		ONCOMING		HIT FIXED OBJ	
		PEDESTRIAN		RIGHT ANGLE	
	CAUSED PERSON TO DODGE	SAME DIRECTION	2 - LANE	SIDESWIPE	
PEDESTRIAN		3 - LANE	REAR END		
DRIVER		4 - LANE	INTERSECTION		
		JUST MISSED ACCT	4 - LANE DIVIDED	HEAD ON	
				RAN OFF ROAD	
15. REMARKS					
16. NAME OF PERSON ISSUING TRAFFIC TICKET					
17. ORGANIZATION AND INSTALLATION				18. RANK / GRADE	

TICKET  
NUMBER  
  
**N**  
**10048354**

DD Form 1408, DEC 87

Previous edition is obsolete.

CO of violator or appropriate civil agency

1

Figure B-17. Ticket Form





## **APPENDIX C. TEST INFORMATION**

### **A. TEST REQUIREMENTS**

1. The application shall be able to generate a Suspension Letter (Figure B-2).
2. The application shall be able to generate a Letter of Acknowledgment (Figure B-3).
3. The application shall be able to generate a Docket (Figure B-4).
4. The application shall be able to generate an Incident Report (Figures B-5, B-6, B-7).
5. The application shall be able to generate a Desk Journal (Figure B-8).
6. The application shall be able to generate a Suspension Report (Figure B-9).
7. The application shall be able to generate a Ticket Analysis (Figure B-10).
8. The application shall be able to generate a Citation Listing (Figure B-11).
9. The application shall be able to generate a Probation Report (Figure B-12).
10. The application shall be able to generate a DUI Listing (Figure B-13).
11. Each report shall be displayed on screen by default with an option to send it to the printer.
12. All forms used for data entry shall be based on existing documents .
13. All reports shall be based on existing documents .
14. The application shall have a graphical user interface.
15. Traffic Administrators shall have access to ticket, browse, and report functions.
16. Traffic Administrators shall have access to registration data to enter in personnel that are not currently registered but have a ticket issued.
17. Registration Clerks shall have access to registration and browse functions.

18. Desk Clerks shall have access to the I/C reports, Desk Journal, browse, and reports functions.
19. Casual Users shall have access to the browse functions.
20. Chiefs shall have access to browse and validation functions.
21. Armorers shall have access to weapons registration and browse functions.
22. System Administrators shall have access to system administration functions.
23. Users shall be able to view related data while browsing through personnel records.
24. Traffic Administrators shall be able to enter ticket information based on a SSN or DOD decal.
25. Each user shall have an individual password and user classification.
26. The user's classification shall determine which functions are available.
27. Passwords shall be between seven to ten case-sensitive alphabetic characters and must be changed every four weeks.
28. The system shall interface with a Scantron model 8200 optical mark reader to enter person and/or vehicle registration data.
29. The browse function shall allow users to perform complex queries that include AND and OR conditions, partial identifiers, and multiple table joins.
30. Traffic Administrators shall maintain the validation tables for Traffic citation numbers, State abbreviations, and the Disposition Indices.
31. The system shall allow cascade archival of personnel and all child records when that person checks out on PCS orders.
32. A person shall have a SSN to be entered in the database.
33. The database shall allow a SSN to be of the form ###-##-####, XXX-YY-####, or CCC-YY-Q###. # represents a numerical digit. X is a literal character, YY

is the current year, ##### is a serialized number, CCC is a country code for foreign students, and Q#### is a number where Q is the quarter in which the foreign student reports and the ### is a serialized number.

34. SSN's of the form XXX-YY-#### shall be provided by the application when necessary to register a student that does not have a SSN already issued.
35. A ticket shall be issued to a person that is in the database.
36. A vehicle shall belong to a person in the database.
37. A ticket shall have a disposition Index.
38. The application shall not be accessible without a valid login.
39. The table containing user Ids and Passwords shall be encrypted.
40. The application shall be run on a network with up to 75 simultaneous users.
41. System Administrators shall not be able to delete their own account.
42. Tickets shall be able to be deleted or re-associated with a different person than to whom it is already issued.

## **B. TEST CASES**

The following section steps through two test case aggregations. They list: the required data; the test procedure; and the desired results. In the listing of desired results, requirements related to the result are listed in parenthesis.

### **1. Aggregation One: Inspection**

This test case verifies inspection requirements 27, 32,33, 35, 36, 37, and 39. There is no data required to test these requirements, only the table structures that the data is stored in.

The testing process is to use the Paradox DBMS to inspect the user, person, ticket, and vehicle table structure to ensure the appropriate validity constraints are in place.

The desired results are:

- The system will prompt the tester for a password before allowing access to the user table (39).
- The “picture” of the password field within the user table is of the form “\*7{@}[\*3{@}]” (27).
- The SSN field in the person table is a required field (32).
- The SSN field in the person table has a “picture” that allows a SSN to be of the form ###-##-####, XXX-##-####, &&&-##-#### (33).
- The index field in the ticket table is a required field (37).
- The ticket table has a referential integrity constraint as a child with the person table as the parent (35).
- The vehicle table has a referential integrity constraint as a child with the person table as the parent (36).

## 2. Aggregation Two: Execution

This aggregation verifies requirements 1, 2, 3, 6, 7, 9, 10, 12, 13, 14, 15, 16, 23, 24, 34, and 42. The required data is the application, all the previously converted data files as outlined in Chapter 4 and a Traffic Administrator account. The testing procedure is:

- Login as a Traffic Administrator. Observe the functions available from the main menu.
- Observe the interface; is it a GUI or command line?
- Enter the browse function, choose a personnel record at random, view all related information. Write down vehicle decal number and person’s SSN.
- Return to main menu. Enter ticket function.
- Select parking ticket option, and press new button.
- Enter decal number from previous step when prompted.

- Verify personal information that appears on the screen observe the format of the scree. Press next page button and view the format of the screen on this page.
- Enter a ticket for this person (write down the ticket number).
- Return to ticket menu by pressing the done button
- Select the moving violation option and press the start button
- Enter the SSN written down in the previous step when prompted.
- Enter another ticket for this person (write down the ticket number). Include a court date for this ticket (write down the court date).
- Return to ticket menu by pressing the done button.
- Select the new record option and press the start button.
- Observe the blank record that appears, press the generate SSN button, observe the format of the SSN that appears in the SSN field.
- Enter a ticket for this person (write down the ticket number and the assigned SSN) and return to the ticket menu.
- Press the suspension button, press the letter button, enter the generated SSN written down from the preceding step when prompted. Edit the form letter that appears. Press the next button to edit paragraphs two and three. Press the acknowledge button to edit the letter of acknowledgment.
- Press the print button located on the letter of acknowledgment page (application will return to suspension dialog box).
- Press the new button. Enter the generated SSN written down from the previous step when prompted. Also enter a start date, end date, and type. Press done button to return to suspension menu. Press done button to return to ticket menu.
- Enter the fictitious person (generated SSN), into the probation list by pressing the probation button, pressing the new button, and filling in the requested information. Press the done button to return to the probation menu then done to return to the ticket menu.

- Enter the fictitious person (generated SSN), into the DUI list by pressing the DUI button, pressing the new button, and filling in the requested information. Press the done button to return to the DUI menu then press the done button to return to the ticket menu.
- Press the done button to return to the Traffic Administrator's main menu.
- Press the reports button.
- Press the Docket button.
- Press the suspension button.
- Press the DUI button.
- Press the probation button.
- Press the statistics button. Compare results with manually calculated reports.
- Press the done button to return to the Traffic Administrator's main menu.
- Press the ticket button to enter the ticket main menu.
- Press the modify button. Enter the test ticket-number.
- Press the next page button, locate the test tickets press the delete button.
- Log out of application
- View reports for format and content.

The desired results are:

- The application's interface is a GUI (14).
- The Traffic Administrator has access to ticket, browse, and report functions(15).
- The application generated a suspension letter with the appropriate entered data in the proper format (1, 13).

- The application generated a letter of acknowledgment with the appropriate entered data in the proper format (2, 13).
- The application generated a Docket with the appropriate entered data in the proper format (3, 13).
- The application generated a suspension report with the appropriate entered data in the proper format (6, 13).
- The application generated a probation report with the appropriate entered data in the proper format (9, 13).
- The application generated a statistics report with the appropriate entered data in the proper format (7, 13).
- The application generated a DUI report with the appropriate entered data in the proper format (10, 13).
- The forms used for data entry (ticket form, letters of suspension and acknowledgment) matched existing forms (12).
- Traffic administrator created new record with generated SSN of the format XXX-##-#### (16, 34).
- Traffic Administrator is able to enter a new ticket based on SSN or decal number (24).
- Traffic Administrator is able to view pet, ticket, weapon, vehicle, faculty decal, and storage information while browsing through personnel records (23).
- Traffic Administrator is able to delete tickets that do not belong (42).





## LIST OF REFERENCES

- BOEH81 Boehm, Barry W., *Software Engineering Economics*, Prentice Hall Inc., 1981.
- ELMA94 Elmasri, Ramez and Navathe, Shamkant B., *Fundamentals of Database Systems*, Benjamin Cummings Publishing Company Inc, 1994.
- NAVA91 Naval Computer and Telecommunications Station, *Users Manual: Computerized On-Line Police System (COPS)*, 1991.
- SOMM92 Sommerville, Ian, *Software Engineering*, Addison Wesley Publishing Company, 1992.



## INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 013 Naval Postgraduate School Monterey, California 93943-5101	2
3.	Timothy Shimeall, Code CS/SM Naval Postgraduate School Monterey, California 93943	2
4.	John Falby, Code CS/FA Naval Postgraduate School Monterey, California 93943	2
5.	NPS Base Police Dept, Code 44 Naval Postgraduate School Monterey, California 93943	2
6.	Lawrence A. Nathan PO Box 967 Quantico, Virginia 22134	2
7.	Albert Nathan 8950 E. Bannister Rd. Kansas City, Missouri 64134	2
8.	Director, Training and Education MCCDC, Code C46 1019 Elliot Rd. Quantico, Virginia 22134-5027	1